PART-B PROGRAMS

Develop a javascript to sort and accessing the array elements.

Input: n-5

Unsorted array elements: 10,1, -5,4,15

Sorted Array: 5,1, 4, 10, 15

<html>

<head>

<title>Array Sorting</title>

</head>

<body>

<script>

var inputArray = [];

var size = 5; // Maximum Array size for (var i=0; i < size; i++) {

InputArray[i] = parseInt(prompt('Enter Element (i+1)));

1

document.write("Unsorted array elements:" + inputArray); document.write("<br><br>");

let sortedArrayInputArray.sort((a, b) a-b);

document.write("Sorted array elements:" sortedArray);

</script>

</body>

</html>

Create a class by the name rectangle with 2 attributes length and breadth. Include parameterized constructor to assign values to data members and a function to calculate area of the rectangle. Demonstrate creation of object of class rectangle and display ita area

Input: length 3, breadth 6 Output: Area-30

<!DOCTYPE html>

<html>

<head>

<title>flectangle Class</title>

</head>

<body>

<script>

class Rectangle{

constructor(length, breadth) {

this.length length;

this breadth breadth;

}

calculateArea() {

return this.length\*this breadth;

}

}

const rect = new Rectangle(5, 6);

document.write("Input: length="+rect.length + ", breadth=" + rect.breadth+"<br>");

document.write("Output: Area="+rect.calculateArea());

</script>

</body>

</html>

Develop a JavaScript to demonstrate the working of callback and async functions.

<!DOCTYPE html>

<html>

<head>

<title>Callback and Async Functions Demo</title>

</head>

<body>

<script>

function fetchData(callback) {

setTimeout(() => {

const data = "Async data received!";

callback(data);

}, 2000);

}

function processAsyncData(data) {

console.log("Processing data:", data);

}

fetchData(processAsyncData);

async function fetchDataAsync() {

return new Promise(resolve => {

setTimeout(() => {

const data = "Async data received using async function!"; resolve(data); }, 2000); });

}

// Async function to process the data async function processDataAsync() {

try {

const data = await fetch DataAsync(); console.log("Processing data (async):", data);

} catch (error) { console.error("Error fetching async data:", error);

}

}

processDataAsync();

</script>

</body>

</html>

Develop an arrow function in JavaScript that checks whether a year is leap year, alert the user with true if the year is leap year and false if year is non leap year. Validate centuries also

Input: 2000, Output: Leap year

Input: 2100, Output: Non-Leap year

Input: 2004, Output: Leap year

Input: 2006, Output: Non Leap year

<!DOCTYPE html>

<html>

<head>

<title>Leap Year Check</title>

</head>

<body>

<script>

const isLeapYear year> {

// Check if the year is divisible by 4

if (year % 4= 0) {

if (year % 1000 && year % 400 == 0) {

return false; // Not a leap year

} else {

return true;

}

} else {

return false; // Not a leap year

const alertLeapYear = () => {

const checkYear = prompt('Enter a year:');

const result isLeapYear(parseInt(checkYear));

if(result){

alert(checkYear + " is a leap year");

}else {

alert(checkYear + " is not a leap year");

}

};

alertLeapYear();

</script>

</body>

</html>

Develop a JavaScript that accepts length and breadth of rectangle as parameter of an arrow functions. Call the function using spread and rest operator and alert the user with perimeter of the rectangle.

<!DOCTYPE html>

<htmlb

<head>

<title>Rectangle Perimeter</title>

</head>

<body>

<script>

//Arrow function to calculate the perimeter of a rectangle

const calculatePerimeter (dimensions) => {

const [length, breadth] dimensions:

return 2 \*(length + breadth);

};

const alertPerimeter() {

const length = parseFloat(prompt("Enter the length of the rectangle:"));

const breadth = parseFloat(prompt("Enter the breadth of the rectangle:"));

const dimensions = [length, breadth];

const perimeter calculate Perimeter(...dimensions);

alert("The perimeter of the rectangle is: "+ perimeter);

};

alertPerimeter();

</script>

</body>

</html>

Develop a JavaScript to demonstrate the usage of optional and default parameters in a

function.

<!DOCTYPE html>

<html>

<head>

<title>Opt def parameters</title>

</head>

<body>

<script>

//Function with optional and default parameters

function greet(name, greeting = "Hello") {

if (name=== undefined) {

console.log(`${greeting}, there!`);

} else {

console.log(`${greeting}, ${name}`);

}}

greet("Alice"); // Outputs: Hello, Alice!

greet("Bob", "Good morning"); // Outputs: Good morning, Bob!

greet();

</script>

</body>

</head>

Create a class by the name box with parameters length, breadth, and height. Create a class boxweight that extends box and include a new parameter weight. Create another class by the name boxcost that extends boxweight and has a parameter by the name shipmentcost. Include constructors in all the classes. Create an object of boxcost and display values of all parameters that represent multilevel inheritance.

<html>

<body>

<script>

class Box {

constructor(length, breadth, height) {

this.length = length;

this.breadth breadth;

this.height height:}}

class BoxWeight extends Box {

constructor(length, breadth, height, weight) {super(length, breadth, height);

this.weight weight;}}

class BoxCost extends BoxWeight {

constructor(length, breadth, height, weight, shipmentCost) {

super(length, breadth, height, weight);

this.shipmentCost shipmentCost;

}}

const myBoxCost new BoxCost(10, 8, 5, 3.5, 20);

document.write("Box Dimensions:", "length=", myBoxCost.length,’’ breadth=”myBoxCost.breadth, "height=", myBoxCost.height);

document.write("<br><br>");

document.write("Box Weight: ", myBoxCost.weight);

document.write("<br><br>");

document.write("Shipment Cost:", myBoxCost.shipmentCost);

</script>

</body>

</html>

Program 8

MY SQL

CREATE DATABASE student db;

SHOW DATABASES;

USE student db;

CREATE TABLE students (

id INT AUTO INCREMENT PRIMARY KEY, studentname VARCHAR(255) NOT NULL,

usn VARCHAR(255) NOT NULL );

INSERT INTO students (studentname, usn)

VALUES

('Jane', '789012'),

('Alice', '345678'),

('Bob', '901234');

select \* from students

SERVER.JS

const express = require('express');

const mysql = require('mysql2');

const cors = require('cors');

const app = express();

app.use(cors());

const port = 3000;

const db = mysql.createConnection({

host: 'localhost',

user: 'root',

password: 'password123', database: 'student db' });

db.connect((err) => {

if (err) { throw err; }

console.log('Connected to MySQL'); });

// API endpoint to fetch all students

app.get('/students', (req, res) => {

db.query('SELECT \* FROM students', (err, result) => {

res.json(result);

if (err) throw err; }); });

app.listen(port, () => {

console.log('Server is running on port ${port}'); });

INDEX HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale-1.0">

<title>Student Database</title>

</head>

<body>

<h1>Student Database</h1>

<div id="studentList"></div>

<script>

fetch('http://localhost:3000/students')

then(response response.json())

then(data => {

data.forEach(student => {

studentList.innerHTML += '<p>Name: ${student.studentname), USN:

$(student.usn)</p>';

});

})

.catch(error=>{

console.error('Error fetching data: ', error);

});

</script>

</body>

</html>